Retinal ganglion cell software and FPGA model implementation for object detection and tracking

Diederik Paul Moeys[1], Tobias Delbrück[1], Antonio Rios-Navarro[2] and Alejandro Linares-Barranco[2]
[1]Institute of Neuroinformatics, University of Zürich and ETH Zürich, Switzerland
[2]Robotic and Technology of Computers Laboratory University of Seville, Spain

***Abstract*—This paper describes the software and FPGA implementation of a Retinal Ganglion Cell model which detects moving objects. It is shown how this processing, in conjunction with a Dynamic Vision Sensor as its input, can be used to extrapolate information about object position. Software-wise, a system based on an array of these of RGCs has been developed in order to obtain up to two trackers. These can track objects in a scene, from a still observer, and get inhibited when saccadic camera motion happens. The entire processing takes on average 1000 ns/event. A simplified version of this mechanism, with a mean latency of 330 ns/event, at 50 MHz, has also been implemented in a Spartan6 FPGA.**

I. Introduction

To increase power efficiency, decrease data rate and latency, neuromorphic sensors have been developed over the last 30 years. The Dynamic Vision Sensor (DVS) [1] is an example of such category of devices which draws inspiration from the real functioning of the retina. This vision sensor outputs temporal contrast of logarithmic intensity, asynchronously, through Address Event Representation (AER). The AER protocol encodes the x-y address of where the change happened, to which a microsecond timestamp is added. The in-pixel processing imitates the inherent processing of the Retinal Ganglion Cells (RGC) in biological retinas with microsecond resolution, allowing tasks such as high-speed tracking [2]. Therefore, to investigate further the efficient processing of visual information of the brain, this work tries to mimic further the pre-processing intrinsic to the retina, computed by a specific type of RGC, described in [3]. The Object Motion Cell (OMC) detects local motion by getting excited by small moving objects and being inhibited by large synchronous global motions of the scene (saccades). This feature can be used to extract basic information about motion of a target object. A simulation of an analog imager based on the OMC was previously attempted in [4] but was centered around a different model and technology.

II. The Circuit Model

A. The observed neural mechanism

The mechanism of the OMC is possible due to its excitatory-inhibitory center-surround morphology. Although the Receptive Field (RF) of the cell is in practice more like a two-dimensional (2D) Laplacian function, it can be simplified to a 2D top-hat function, with positive weight in its center and negative on its outside. The basic algorithm on which this cell is based is summarized in Fig. 1 and is reported in the following steps. The RF of the cell is composed of subunits of similar sizes: these represent the single bipolar cells of the retina. The central subunits are excitatory while all the other subunits are inhibitory. Since the bipolar cells are not inhibitory themselves, their inhibition is theorized to be mediated by fast amacrine cells [3]. When a change in brightness is detected somewhere in the RF of the cell by a hyperpolarizing cone, the membrane potential of the bipolar cell connected to it is increased linearly. Then, a non-linear rectifying transformation is applied to it. While the subunits are integrating they also decay due to an ionic leakage, adapting to the present visual situation. The RGC contacting the bipolar cells of the exciting center and the inhibiting amacrine cells then integrates the net synaptic input (the difference between excitation and inhibition) on its own membrane. If this is higher than its response threshold, the cell fires. The cell works such that if there is a perfectly synchronous motion in the inhibitory surround, the center excitation is cancelled. Otherwise if the excitatory subunits are triggered and not compensated for, the cell spikes.

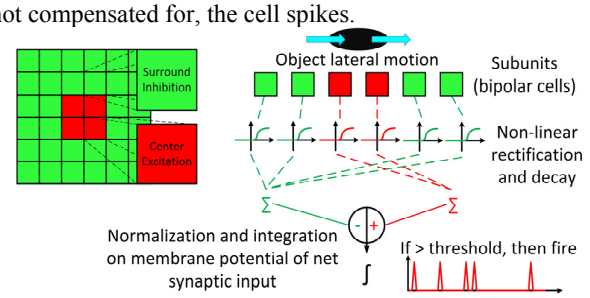

Fig. 1 Object motion cell's simplified computation.

B. Single jAER OMC implementation

To mimic the behavior of a simple OMC, an algorithm has been developed in jAER, the software which processes the events of the DVS neuromorphic sensor [5]. To create the subunits of the OMC, these are set by subsampling the address of the incoming events, making it possible to scale the size of the single subunits by a power of two, to better fit the size of the object to be detected. Then all subunits are set to be inhibitory apart from the central four which are excitatory. When an event is received at a particular x, y coordinate within a subunit, its membrane potential is increased linearly by one unit and its non-linearity is calculated. This non-linear rectification can be set to be, in the developed model, of exponential type of any order or of exponential tangent type. In the first case, a clipping is artificially placed on the membrane potential of the single subunit, so that if too much activity is registered, a single subunit cannot constantly dominate over other weak subunits. In the second case, the natural saturation

This research is supported by the European Commission project VISUALISE (FP7-ICT-600954) and by the Spanish grants (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02) and the Andalusian Council Excellence grant MINERVA (P12-TIC-1300). We thank the Sensors group at INI and the RTC lab in Seville.

of the exponential tangent already inherently performs this operation. To model the ionic leakage which makes the subunit's membrane potential decay if no further activity is detected in its RF, an adjustable exponential decay with time constant τ_s is computed at every event timestamp received and applied to the subunit.

For the whole array of subunits, the total surround inhibition is computed by adding the non-linearized membrane potentials of each subunit and by normalizing. The same is done for the center excitation. The excitation can be scaled by a synaptic weight α. This empirically ensures stability and allows the OMC to be adjusted to different visual scenes. The final membrane potential of the RGC is computed by integrating the net synaptic input and, if larger than the adjustable threshold, the cell fires, signaling object motion detection. The RGC's membrane potential is also decayed exponentially with time constant τ_n. If the exponential tangent non-linearity is chosen then the computation of the OMC can be modelled as:

$$\begin{cases} if\ \ V_m e^{-t/\tau_n} \le V_{IF} \ \ \ do\ not\ fire \\ \ \ \ if\ \ V_m e^{-t/\tau_n} > V_{IF} \ \ \ fire \end{cases} \quad (1)$$

Where V_m is the integrated membrane potential of the OMC expressed in equation (2), t is the timestamp time, V_{IF} is the Integrate and Fire (IF) threshold to be overcome to spike.

$$V_m = \int_0^t \left(\alpha \frac{\sum_{i=1}^{4} tanh\left(V_{ex_i} e^{\frac{-t}{\tau_S}}\right)}{4} - \frac{\sum_{i=1}^{k-4} tanh\left(V_{in_i} e^{\frac{-t}{\tau_S}}\right)}{k-4} \right) dt \quad (2)$$

Where α is the weight of the excitation, V_{ex_i} and $V_{\mathrm{in}_i}$are the ith excitatory and inhibitory subunit membrane voltages respectively and k is the total number of subunits. As an option, the total inhibition (or excitation) can be computed, not only by the membrane potential but by its difference to the neighboring subunits. This local normalization removes the problem of global dimming, which could trigger the RGC response.

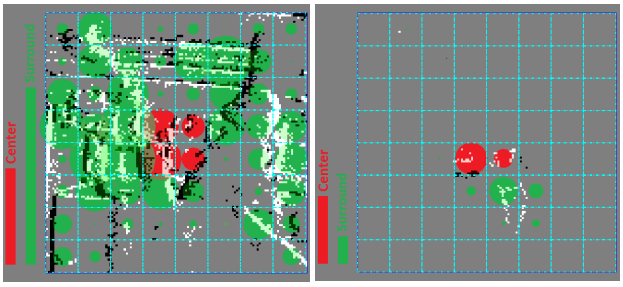

Fig. 2 jAER output showing the DVS camera output events (ON in white, OFF in black). Left: saccade inhibiting the OMC. Right: single object movement exciting it. The green disks represent the activation of the inhibitive subunits and the red ones represent the excitation. The bars on the side compare the normalized activities. The positions of the subunits are highlighted by a grid.

The simple OMC was initially tested in jAER with 8 x 8 subunits. The algorithm works for different natural visual stimuli such as the ones shown in Fig. 2. On the left image, the DVS camera output shows a saccade view of the office. On the right side, only a person moves in the scene. As it can be seen, when the camera is moved in a saccade, the inhibition subunits are maximally active (green disks) and compensate for the excitatory input (red disks). If the local motion at the center of the RF is instead not compensated, then the cell fires. This can be seen in the graph of Fig. 3, which illustrates a plot of center excitation and surround inhibition with a second order non-linearity added to each subunit.

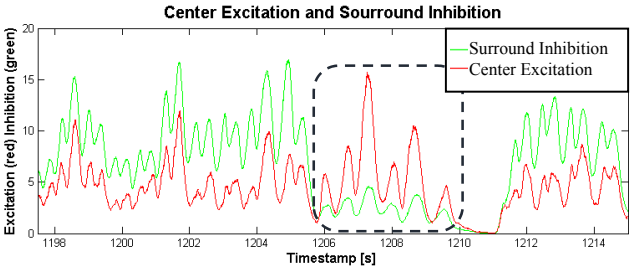

Fig. 3 Plot of surround inhibition versus center excitation for the video of Fig. 2 in arbitrary units. The non-circled parts correspond to global motion (excitation smaller than inhibition). The circled part corresponds instead to local motion (excitation larger than inhibition).

III. Multiple OMC Tracker jAER Implementation

A. Multiple jAER OMCs implementation

To make use of the OMC for tracking, it is important not to just detect the object's presence but also its direction. To achieve this goal, an array of 16 x 16 subunits was set up in jAER. By sliding the 2 x 2 excitation center of the OMC across all subunits with unity stride, 15 x 15 OMCs (this number is dictated by the size of the target) can be constructed. To speed up computation, all subunits are regarded as the inhibiting surround, including the central ones, so that the total inhibition can be computed just once for all OMCs with minimal error. This would mean that the term $k-4$ in the right hand side of equation (2) simplifies to k. The final result is that now all these overlapping cells respond to object motions at specific locations. To make the algorithm more robust, the average event rate is used. For the DVS128 sensor, if this is lower for example, than 500 ev/s (events/second), then the IF threshold of the RGC can be set to a high value. This is because for such low event rates the pixels producing a response constitute mostly random activity due to leakage in the reset transistor of the DVS pixel [1]. Increasing the threshold prevents the cell from firing for no relevant activity. An upper threshold can also be set if the activity is too high: in such case it is likely that the sensor either moves very close to a high contrast wall or that the target is too close and covers the entire field of view. In this case, tracking is not necessary and can be suppressed. Any event rate in between these boundaries can be associated with a moving object. The event rate numbers used in this design have been obtained empirically by placing a 128 x 128 DVS sensor on a moving robotic platform following another robot.

B. Tracking

The single OMCs spiking in the presence of a moving object can be easily clustered, so as to obtain the position of the moving object by correlation. A tracking scheme was implemented to draw a containing box around the last 3 spiking cells close in time and space, and to find its center of mass by geometry. If some OMCs spike due to a second object moving in the scene, therefore beyond the reach of the first tracker, then

these outputs can be associated to a second tracker. Double tracking is shown in the left image of Fig. 4, for a still camera staring at the scene where two objects are moving independently. The trackers reset automatically when no OMC near them spikes within an adjustable amount of time, and reappear at another location where an object seems to be moving. This way the tracker can be reused once the first object stops moving and the reset event can trigger the memorization of its last known position.

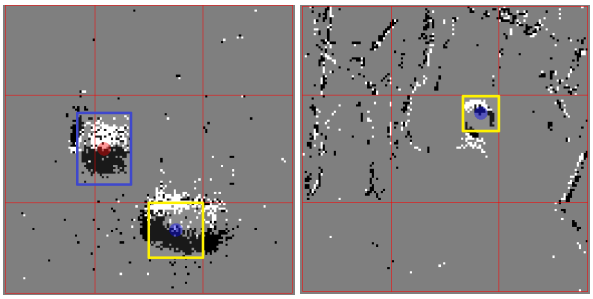

Fig. 4 15 x 15 array of OMCs: on the left, two objects activate both the trackers (yellow and blue with respective blue and red center of mass), on the right the moving object can still be tracked while the camera is moving in the scene thanks to excitation weight adjustment. The position of the center of mass with respect to the 9 red quadrants can be used by a robot to plan its next action.

By dividing the field of view in 9 quadrants and knowing the position of the center of mass of the tracker within one of them, it is possible for the robot carrying the sensor to plan its next move and follow, or just shoot, the target. The size of the target can also be roughly estimated with a very simplified inverse pinhole camera model in the x direction knowing the width of the target object and ignoring lenses' distortion. The numbers need to be heavily low-pass filtered in order to obtain a reliable measurement, but the order of magnitude of the result is at least consistent with the ground truth.

Since the algorithm works in such a way that the OMC gets inhibited in the case of global motion, the same happens when the observer is moving and tracking gets suppressed. To still allow the OMCs to fire and continue tracking even while the observer is moving at moderate speed, parameters need to adapt. An approach to solve this problem effectively is to increase the weight of excitation α by a fixed amount when a certain activity, indicating apparent scene motion, is detected. This way the firm movement of an object can still cause the OMCs to fire even though inhibition is stronger.

This algorithm works however only if the apparent motion of the scene is slower than the one of the target object. Also, if corners or high-contrast features suddenly appear in the field of view, these might be detected as objects to be tracked. Only the temporal and spatial correlation of the OMCs spiking which are part of the tracker can guarantee that the correct object is still followed. This can be seen in the right image of Fig. 4.

IV. FPGA IMPLEMENTATION

A. Hardware used and multiple OMC FPGA implementation

The FPGA design was approached to explore the possibilities of implementing complex cell types in logic to exploit parallelism. The OMC mechanism was prototyped in the Spartan6 XC6S1500FXT Xilinx FPGA board developed in [6], called AERNode board. The design of this platform allows multi-board communication with conventional parallel-handshake-AER chips, serial Low-Voltage Differential Signaling (LVDS) connections or robots with the adequate motor interfaces. A daughter board based on an OpalKelly module, called OKAERTool, is used for monitoring, or sequencing, and logging, or playing, events from and to the AERNode board. It is in fact able to sequence events from its on board DDR2 128MB SDRAM to the AERNode board and to monitor its output through USB2.0 in jAER. OKAERTool is fundamental for debugging the design implemented in FPGA.

Due to the limited number of resources of the FPGA available (gates and memory), the OMC implementation strategy was changed into a much more simplified one. Five OMCs' centers were fitted in the center of each quarter of the field of view and in its center to obtain the most basic directions of a moving object. Every OMC center consists of 2 x 2 subunits. The design consists of two levels: a Mother Cell (MC) which deals with the four-phase AER handshake protocol (request and acknowledge) with the outside neighboring blocks and five inner Daughter Cells (DC) which, in parallel, each calculate the excitation of a particular OMC. The MC calculates the inhibition (set to be the entire field of view, as in the jAER model) and feeds it along with the incoming request to the DCs if the input event falls within their excitation center. The MC also manages the global high-priority decay of the subunits by a counter. The DCs then propagate their request, in case of firing, to the following stage through the MC. Since the cells all work in parallel and store their firing in a one-hot coded output vector, the processing delay does not scale with the number of DCs active. The active low requests are anded so that the request to the next stage is active for at least one DC active.

The algorithm of each single DC follows the most basic jAER implementation, however, to reduce the use of resources variables are restricted to 16 bit values and every operation is simplified. Since divisions (for normalization of inhibition and excitation) are performed by even numbers, these are done by multiples of 2 by bit-shifting. The same bit-shift operation substitutes the non-linearity and a saturation is achieved with a comparator when a certain value is attained. The global decay is also achieved with bit-shifts and it is adjustable by the counter's limit. Finally, only one multiplication is present, the one for the integration of the single daughter OMCs' membrane potential. Three parameters can be set via Serial Peripheral Interface (SPI) through the OKAERTool. These are the IF threshold, the decay counter's limit and the excitation weight α.

B. System integration and hardware

The system of OMCs, enclosed by the MC, was integrated with a pre-existing system architecture. This was achieved by creating a separate, parallel processing branch through a splitter and a merging element. This can be seen in Fig. 5. The input request and parallel data which the OMC receives are the same that the cascade of filter elements receives (in this case the Hot Pixel Filter, which filters out addresses of pixels with high spike rate). In case of firing, the OMC sends its request further to a

merging arbiter along with its output data (the one-hot-coded firing DC's address). The arbiter decides randomly which branch will be serviced first and acknowledges it after reading its data. The acknowledge signals of both branches propagate back to a Muller C-element latch which combines them into a single one which is then sent back to the event source (the DVS or the event sequencer).

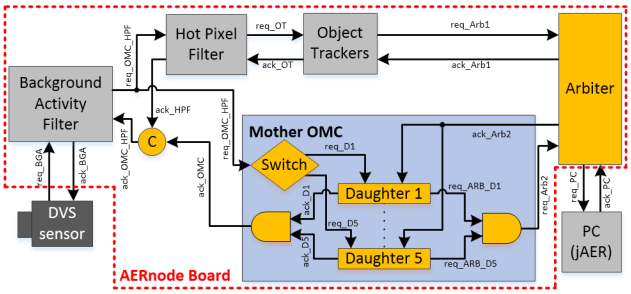

Fig. 5 Integration of the OMC with the existing architecture of [7].

A problem which was encountered during design was that if both branches would be requesting at the same time, the arbiter would be waiting for the request withdrawal of the branch requesting first, before servicing the second branch. The request withdrawal cannot happen until the acknowledge signal of the branch is propagated back to the event source. Since, due to the C-element, this cannot happen until both acknowledge signals are received, the system effectively deadlocks. To overcome this problem caused by the dependency of the merger from the splitter, a feature of collision detection was added to the arbiter: the latter can in fact now acknowledge the two branches one after the other even though the request of the first is not yet de-activated. The implementation of Fig. 5 for 5 DCs uses 4% of the slice registers, 11% of the slice Look-Up Tables (LUT) and occupies 16% of the available slices. For 9 DCs, the resource consumption changes to 5%, 12% and 19% respectively.

V. Results

A. Latency and power consumption comparison

To estimate the delay of the OMC tracker in both jAER and FPGA, the time taken for an input event to be processed was measured. For the jAER OMC tracker, the nanoTime() method of class System was used to measure the processing time for 3 different numbers of OMCs (though the difference between 5 and 9 cells is irrelevant). It should be noted, however, that this delay increases by 25-30% if the input events are not being read directly from the hard disk (as it is the case for the reported numbers) but if they are obtained in real-time from the DVS sensor. This is because events are processed at a higher speed (at the maximum of the system's capabilities) if read from a logged file rather than if the data is obtained from the real world. For the FPGA performance, the numbers were obtained using the Xilinx ChipScope tool, since the processing delay was below the microsecond event timestamping resolution. The results are summarized in TABLE I. The OMC in FPGA takes 22 and 11 clock cycles, depending if the incoming event falls in one of the DC's RF, to process an input event and complete the AER handshake with a 50 MHz clock. As regards the power consumption comparison, the FPGA has a factor of at least 100 of advantage over even the small embedded Intel Next Unit of Computing (NUC).

VI. Conclusion

This paper offers two implementations of the OMC model for the purpose of object detection and tracking: one software-based and FPGA-based. A comparison study is presented between these two systems and highlights the power consumption and latency advantages of the FPGA. Due to its parallelism, the nanosecond latency does not scale with the number of OMCs implemented. This paper however does not yet present the next step: the recreation of the 15 x 15 OMCs tracker of jAER into FPGA. This is because the resources of the Spartan6 constrain for the moment the OMC array size. Knowing that the number of occupied slices for the 5 OMCs design is 16% and for the 9 OMCs design this is 19%, it can be approximately estimated that at most another 108 OMCs, each taking 0.75% of the resources, can be fitted into the current design (still 117 below the desired 225). The work to come will therefore focus on further reducing the size of the DCs and on the optimization in the integration with the existing architecture. At the moment, the output of the OMCs is already used to choose the location where to initialize the trackers of [7] and to validate their operation: the object tracker which is active can now only exist if its center of mass location falls within the RF of a DC firing.

TABLE I. Specification Table of the FPGA OMC

Specification Table			
	jAER (64-bit Intel NUC, 4 GB RAM, i5-4250U, 1.30 GHz)	jAER (64-bit PC, 16 GB RAM, i7-4770K, 3.50 GHz)	FPGA (Xilinx Spartan6, 50 MHz)
Latency of 5x and 9x OMCs	~500 ns/ev at 0.2 Mev/s, at CPU load < 5%	~250 ns/ev at 0.2 Mev/s , at CPU load < 2%	220 or 440 ns/ev at any event rate
Latency of 15x15 OMCs	~1000 ns/ev at 0.2 Mev/s , at CPU load < 5%	~500 ns/ev at 0.2 Mev/s , at CPU load < 2%	NA
Power	6.2 W static + 6.2 W for running jAER + 2.48 W for 5/9x OMCs or 3.72 W for 15x15 OMCs	A few hundreds of Watts	0.775 W static + 0.05 W for 5/9x OMCs

References

[1] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128 x 128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE J Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[2] T. Delbruck and M. Lang, "Robotic Goalie with 3ms Reaction Time at 4% CPU Load Using Event-Based Dynamic Vision Sensor," *Front. Neurosci.*, vol. 7, p. 223, Nov. 2013.

[3] S. A. Baccus et al., "A Retinal Circuit That Computes Object Motion," *J. Neurosci.*, vol. 28, no. 27, pp. 6807–6817, Jul. 2008.

[4] K.-C. Tseng and A. C. Parker, "A neuromorphic circuit that computes differential motion," in *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2012, pp. 89–92.

[5] "jAER Open Source Project," *jAER Open Source Project*. [Online]. Available: http://jaerproject.org. [Accessed: 17-Sep-2013].

[6] T. Iakymchuk et al., "An AER handshake-less modular infrastructure PCB with x8 2.5Gbps LVDS serial links," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1556–1559.

[7] A. Linares-Barranco et al., "A USB3.0 FPGA event-based filtering and tracking framework for dynamic vision sensors," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 2417–2420.